Neurocomputing 449 (2021) 280–289

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

V-SOINN: A topology preserving visualization method for multidimensional data

Hui Dou [a], Baile Xu [a], Furao Shen [a,*], Jian Zhao [b]

[a] State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China
[b] School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China

ARTICLE INFO

Article history:
Received 1 October 2020
Revised 18 January 2021
Accepted 29 March 2021
Available online 2 April 2021
Communicated by Zidong Wang

Keywords:
Data visualization
Self-organizing incremental neural network (SOINN)
Self-organizing map (SOM)

ABSTRACT

Data visualization plays an important role in data analysis by displaying data to observers in an interpretable way. Visualizing multidimensional data requires projecting the data into a low-dimensional space that is visible to humans. In this paper, we propose a neural network model that can generate such projections while preserving the topology relationships within data points, which is named Visible Self Organizing Incremental Neural Network (V-SOINN). V-SOINN is able to construct a topology preserving visible network automatically and classify visible nodes to different classes in the low-dimensional space. The thought of topology preserving visualization stems from Self-Organizing Map (SOM). Compared to SOM, the main advantage of V-SOINN is that it does not need prior decision of network structure, including the number of nodes and grid in the output layer. V-SOINN can show the density distribution of datasets by using the activation counts of datasets. V-SOINN is able to depict the number of classes in the low-dimensional space as well. We perform experiments on artificial and real-world datasets, and V-SOINN outperforms PCA, MDS, t-SNE, Neural Gas and SOM on the datasets. Experiments show that V-SOINN can preserve the topology and V-SOINN can produce the correct classification result when the number of samples is small.

1. Introduction

Nowadays, data generated from almost every field of our life are recorded by computers. These data creates a potential resource library, in which we can explore valuable information. However, it is difficult to find valuable information in high-dimensional data. Combining the computer and human in the data exploration process is an effective approach. Due to the limitation of human visual system, data in a 4-D or higher dimensional space can not be observed directly. Visualizing such data requires projecting the data into a lower dimensional space that is visible to humans.

Mapping high-dimensional data into low-dimensional spaces is the core issue of data visualization [1]. A straightforward method is using pairwise feature combinations. However, it requires n^2 figures to get the full view of an n-dimensional dataset, and the interpretability of each figure declines as the dimensionality increases. Many dimensionality reduction methods can be used for visualization. Principle Component Analysis (PCA) [2] and Multidimensional Scaling (MDS) [3] are linear mapping methods. PCA maximizes the distance of data projected in the low-dimensional space, while MDS tries to keep the Euclidean distance between two points in both the high-dimensional space and the corresponding low-dimensional space. These linear mapping methods suffer from heavy computational costs on large datasets [4]. Non-linear mapping approaches such as Isomap [5], Laplacian Eigenmaps [6], t-distributed stochastic neighbor embedding (t-SNE) [7] are utilized, but they are weak in preserving the local or the global structures of the data [1].

In the past few years, many other effective methods have been proposed to realize the visualization of high-dimensional datasets, such as scatter plots, parallel coordinates and pixel display [8]. A dissimilarity-preserving projection technique is proposed in the [9] that keeps the relationships among the mean values of the ensemble members and the relationships among the distributions of ensemble members. An approach that combines Multidimensional Projection with Parallel Coordinates is proposed to visualize the instances relationship [10]. Takashi proposed a visualization method based on SOM that is for clarifying cluster boundaries of high-dimensional datasets [11].

* Corresponding author.
E-mail addresses: huidou@smail.nju.edu.cn (H. Dou), blxu@smail.nju.edu.cn (B. Xu), frshen@nju.edu.cn (F. Shen), jianzhao@nju.edu.cn (J. Zhao).

https://doi.org/10.1016/j.neucom.2021.03.113
0925-2312/

The essence of high-dimensional data visualization is to explore a projection that presents high-dimensional data in a low-dimensional output space. In the process of projection, the core of projection is to preserve some inherent features of high-dimensional data. The criterion for measuring the quality of a projection is whether it can maintain the inherent characteristics of high-dimensional data. The ability of topology preserving, clustering and dimension correlation preserving can be used to measure a projection approach.

The visualization process can be briefly summarized as follows: a) Projection. Projection is the core stage of visualization methods. Visualization methods project high-dimensional data to low-dimensional output data. There are several commonly used projection approaches, such as PCA, MDS, t-SNE, etc. b) Rendering. In this stage, low-dimensional output dataset is mapped into 1-D or 2-D space. Visual features, such as position, color, size or shape are often used.

In this study, we mainly focus on topology preserving projection and flexible expression. The thought of topology preserving projection stems from Self-Organizing Map (SOM) [12], which is a neural network model based on unsupervised competitive learning. It can be used for visualization of multidimensional data by displaying data points in a low-dimensional lattice [13]. The topological relationship in the data is maintained by projecting adjacent data points into the connecting nodes of the lattice. However, the network structure of SOM must be predefined before learning the input data, and the predefined structure is not always suitable for the input data. For example, some nodes will never be activated during training, and some nodes may correspond to blank areas in the high-dimensional data space. The 2-D node is activated once when a data point is projected to it. No matter how many times a node is activated, it is only one small point in the 2-D space. It is hard to see the density distribution of data points accurately. A well-known incremental SOM like algorithm is called Neural Gas (NG) [14]. NG is a competitive learning neural network. The algorithm adjusts the winner in the presence of an input vector and updates the remaining prototypes according to their similarities to this input vector.

The number of classes shown in the low-dimensional space is a kind of visual information that is important for understanding the input data underlying characteristics. It is necessary for the visualization algorithm to illustrate the classification results in the low-dimensional space.

We propose a new algorithm for visualizing the multidimensional data, which is named Visible Self-Organizing Incremental Neural Networks (V-SOINN). V-SOINN is able to construct a topology preserving visible network automatically in the low-dimensional space. V-SOINN is based on ESOINN [15]. Like earlier SOINN [16] and ESOINN models, V-SOINN is a kind of competition-based learning Neural network model, capable of incremental unsupervised learning. V-SOINN preserves the local topological structure of high-dimensional input. During the training process, the low-dimensional visible nodes are learned incrementally, while preserving the relative distance between the input data points. All the points are projected to visible nodes in the 2-D space while keeping neighboring data points to neighboring visible nodes. The visible nodes are classified by using their connection in the network structure. The size of the low-dimensional visible node is in direct proportion to the count of being activated. The distribution of input data is shown in the low-dimensional space.

Compared with SOM, V-SOINN can avoid the difficulty of predefining network structure. V-SOINN has a flexible expression on the density distribution of datasets. A pairwise Elastic SOM (ESOM) combining the preservation of topographical structure and at the same time the relative distance of the data was proposed in [17]. ESOM finds the best matching unit through SOM and updates positions of the reference vectors while keeping the pairwise distance between the input data points. Compared to ESOM, V-SOINN is based on ESOINN. V-SOINN learns a low-dimensional representative set, and figures out the classification results of the representative set in the 2-D space.

The rest of this paper is organized as follows. In Section 2, PCA, t-SNE, SOM are briefly reviewed. In Section 3, V-SOINN is presented in detail. Experimental results are shown in Section 4. Experiments on artificial and real-world benchmark datasets are conducted to compare the effectiveness of topology preserving in V-SOINN, PCA, MDS, t-SNE and SOM. Then we analyze the V-SIONN classification results in the Visible Network. The results show that V-SOINN outperforms other methods on these datasets in terms of relative standard deviation. A summary of this paper is given in Section 5.

2. Related work

In this section, we review the related work, including the basic ideas of Principal Component Analysis (PCA), T-Distributed Stochastic Neighbor Embedding (t-SNE), Self-Organizing Map (SOM) and Neural Gas (NG).

2.1. Principal component analysis

PCA, the principal component analysis, is one of the most widely used data dimensionality reduction methods. PCA is a classic linear method aiming at mapping high-dimensional features to low-dimensional space. PCA achieves the goal by reconstructing k-dimensional features on the basis of the original features. The k-dimensional features called principal components are orthogonal features. The main idea of PCA is to display the principal components in the low-dimensional space and discard the minor components.

The process of PCA is as follows. The vector x_i in the input dataset X is subtracted by the mean of vector:

$$\phi_i = x_i - \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{1}$$

The covariance matrix C is expressed as:

$$C = \frac{1}{n}\sum_{i=1}^{n} \phi_i \phi_i^T = AA^T \tag{2}$$

where $A = \{\phi_1, \phi_2, \ldots, \phi_n\}$. The eigenvectors and eigenvalues are computed from:

$$A^T A v_i = \lambda_i v_i. \tag{3}$$

k largest eigenvalues as well as their corresponding eigenvectors are selected. The transformation matrix P is formed from these eigenvectors in the row manner. The output data in the low-dimensional space can be calculated by the following equation:

$$Y = PA. \tag{4}$$

2.2. T-distributed stochastic neighbor embedding

T-distributed stochastic neighbor embedding (t-SNE) is a highly flexible nonlinear mapping method. The main idea of t-SNE is to use the conditional probability to represent the similarity of the high-dimensional data, as well as the similarity of the low-dimensional data. If the conditional probabilities of the two dimensions are very close, it means that the data in high-dimensional space have been mapped to the corresponding low-dimensional space.

The Euclidean distance between two data points in the high-dimensional space is converted to a conditional probability as:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/(2\sigma_i^2))}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/(2\sigma_i^2))}. \quad (5)$$

The Euclidean distance between two data points in the low-dimensional space is converted to a conditional probability:

$$q_{j|i} = \frac{\exp(-\|x_i - x_j\|^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2)}. \quad (6)$$

Kullback-Leibler divergence is used to measure the distance between two conditional probabilities. The cost function is given as:

$$C = \sum_i KL(P_i||Q_i) = \sum_i\sum_j p_{j|i}\log\frac{p_{j|i}}{q_{j|i}}. \quad (7)$$

2.3. Self-Organizing Map

The Self-Organizing Map (SOM) is a neural network method based on unsupervised learning. SOM maps the high-dimensional data to units which are located on a regular low-dimensional grid. The number of units n is set manually. w_i is a vector, which is the connection weight of unit c_i. Initially, the connection weights are all set to random values.

SOM adopts the winner-takes-all rule. Each input data x finds the unit that best matches:

$$c = \arg\min_i(\|x - w_i\|) \quad (8)$$

where $i = 1, 2, 3, \ldots, n$ means the number of units on the low-dimensional space. c indicates the best match unit (BMU) on the low-dimensional grid. The connection weight of BMU has the shortest Euclidean distance with the input data x. The connection weight of BMU is updated by the random gradient descent method:

$$w_i \leftarrow w_i + \alpha(x - w_i) \quad (9)$$

where α is learning rate.

2.4. Neural gas

The Neural Gas (NG) is a kind of unsupervised learning algorithm. NG can learn the topological structure in the feature dimension. NG takes a winner-takes-most rule. Neural network not only adapts to the nearest neural node, but also adapts to all neural nodes, and its step size decreasing with the increase of distance sequence. The weights of the neural nodes $w_i \in R^n$ are initialized and all connections between nodes C_{ij} are set to 0. Seclect one input data point x, the weights of the neural nodes are updated depending on the "neighborhood ranking" index of x. For each node i, the "neighborhood ranking" index k_i means i is the k_i-th nearest node to x. The weights of neural nodes are updated as:

$$w_i = w_i + \epsilon \times e^{-k_i/\lambda}(x - w_i), \quad i = 1, \ldots, N \quad (10)$$

where ϵ is the adaptation step size and λ is the neighborhood range. The topological structure is learned by updating the connection between neural nodes.

3. V-SOINN

V-SOINN is an unsupervised learning algorithm. V-SOINN is able to automatically construct the network structure in the 2-D space and preserve the topological structure of the input data during the training. It also figures out the classification results of the input data. V-SOINN learns a set of representative input data and projects these nodes to the visible nodes in the 2-D space incrementally while preserving the relative distance between nodes. And then V-SOINN classifies the visible nodes. The relative distance preserving the characteristics of V-SOINN provides posibility of high-dimensional data visualization.

The major steps are listed as follows:

- Construct the high-dimensional network and its corresponding 2-D network
  In this step, representative nodes in the high-dimensional space are learned from the input data iteratively. Then the representative nodes are projected to the visible nodes in the 2-D space for visualization while keeping the relative distance between nodes. Hence a visible network structure is constructed by the visible nodes iteratively in the 2-D space.
- Classify nodes to different classes
  For each unclassified node $a \in A$, we search all the unclassified nodes that are connected with a. All these nodes are classified as the same class in node a. The visible nodes are marked as the same class as their corresponding high-dimensional nodes.
- Map the high-dimensional data into the visible network
  For each data point, the nearest representative node is found in the high-dimensional space, so the input data point is mapped to the corresponding visible node. The visible node is activated once an input data point is mapped to it. The count of the activations is recorded. The size of the visible node in the 2-D space is proportional to the count. The final figure shows the distribution of visible nodes.

The detailed steps of the V-SOINN are listed as follows:

3.1. Step 1: Construct the high-dimensional network and its corresponding 2-D network

3.1.1. Initialization

Here we consider the problem of projecting the dataset X from a high-dimensional space R^n (data space) to a two-dimensional space. The high-dimensional network structure of V-SOINN is a single-layer network that consists of a set of nodes A and a set of connections $C \subset A \times A$. The visible network structure in the 2-D space consists of a set of visible nodes A'. Each node $a \in A$ in the n-D space has a weight vector $w_a \in R^n$ and its corresponding visible node $a' \in A'$ in the 2-D space has a weight $w'_a \in R^2$. w'_a is set randomly at the beginning.

At the beginning, the node set A is initialized to contain two high-dimensional nodes with weight vectors which are chosen from the input data randomly. $A = \{c_1, c_2\}$ and the weight vectors of $\{c_1, c_2\}$ are $w_{c_1}, w_{c_2} \in R^n$. The visible node set A' is initialized as $A' = \{c'_1, c'_2\}$ and the weight vectors of $\{c'_1, c'_2\}$ are $w'_{c_1}, w'_{c_2} \in R^2$. The connection set C is initialized as an empty set.

3.1.2. Node competition

After initialization, the high-dimensional network is trained through competition-based learning iteratively. For one input data point $x \in X$, the winner node s_1 and the second winner node s_2 are determined by:

$$s_1 = \arg\min_{a\in A}\|x - w_a\| \quad (11)$$

$$s_2 = \arg\min_{a\in A\setminus\{s_1\}}\|x - w_a\|. \quad (12)$$

3.1.3. Update the high-dimensional network structure and its parameters

When the winners s_1 and s_2 are found, x is judged whether it is a new node with the winners using the activation threshold T. If the distance between x and s_1 or s_2 is larger than the activation threshold T_{s_1} or T_{s_2}, like $\|x - w_{s_1}\| > T_{s_1}$ or $\|x - w_{s_2}\| > T_{s_2}$, x is a new node. Add it to A and go to 3.1.2 to process the next point.

For each node $a \in A$, its activation threshold T_a is the maximum Euclidean distance between a and its connected nodes:

$$T_a = \max_{(a,b)\in C} \|w_a - w_b\|. \tag{13}$$

If a is not connected to any node, its activation threshold is the minimum distance between a and other nodes:

$$T_a = \min_{b\in A\setminus\{a\}} \|w_a - w_b\|. \tag{14}$$

If the connection between s_1 and s_2 does not exist, it will be created as $age_{(s_1,s_2)}$ and added to C. $age_{(s_1,s_2)}$ is initialized as 0. s_1 and s_2 are topological neighbors. The ages of all edges connected to s_1 are increased by 1.

The weight vectors of s_1, s_2 are updated as follows:

$$w_{s_1} \leftarrow w_{s_1} + \alpha_1(x - w_{s_1}) \tag{15}$$

$$w_{s_2} \leftarrow w_{s_2} + \alpha_2(x - w_{s_2}) \tag{16}$$

where α_1 and α_2 is the learning rates. According to the experimental experience, we set $\alpha_1 = 1/t$ and $\alpha_2 = 1/100t$.

3.1.4. Noise elimination

After a predetermined number of learning iterations, the network deletes the nodes caused by noise. A high-dimensional node is considered to be created by noise if it is isolated, or has only one connection to other nodes. In other words, the node with one or no topological neighbor is removed. The edge $c \in C$ is removed if its age is too large. When a node is removed from the high-dimensional network, its corresponding visible node is also removed from the 2-D visible network.

Algorithm 1. Step 1: Construct the high-dimensional network and its corresponding 2-D network.

Intialization: set $A = \{c_1, c_2\}$, $C \subset A \times A$, $A' = \{c_1', c_2'\}$, $i = 0$, $ITER_1$, $ITER_2$
if $a \in A$ has connection in C **then**
  set $T_a = \max_{(a,b)\in C} \|w_a - w_b\|$
else
  set $T_a = \min_{b\in A\setminus\{a\}} \|w_a - w_b\|$
end if
repeat
  input:$x \in R^n$, set $i = i + 1$
  $s_1 = \arg\min_{a\in A} \|x - w_a\|$
  $s_2 = \arg\min_{a\in A\setminus\{s_1\}} \|x - w_a\|$
  if the distance between x and s_1 or s_2 is larger than the activation threshold T_{s_1} or T_{s_2} **then**
    $A = A \cup \{x\}$
    continue
  else
    $C = C \cup \{s_1, s_2\}$
  end if
  set $age_{(s_1,s_2)} = 0$
  the ages of all edges connected to s_1 are increased by 1.
  $w_{s_1} \leftarrow w_{s_1} + \alpha_1(x - w_{s_1})$
  $w_{s_2} \leftarrow w_{s_2} + \alpha_2(x - w_{s_2})$
  if $i/ITER_1 == 0$ **then**
    Delete the nodes that are created by noise.
  end if
until All nodes in A are trained.
for all nodes do: $w_a = \frac{w_a - u}{\sigma}, w_a' = \frac{w_a' - u'}{\sigma'}$
$S_1(a) = \sum_{b\in A}(d(w_a - w_b) - d(w_a' - w_b'))^2$
repeat
  repeat
    set $a \in A$
    $w_a' \leftarrow w_a' - \eta_1 \frac{\partial S_1(a)}{\partial w_a'}$
  until All nodes in A are trained.
until $ITER_2$ times

3.1.5. Update the 2-D visible network structure and its parameters

The high-dimensional network is trained completely. Our goal is to make the topological network structure of the high-dimensional space be maintained in the 2-D space. If the data point x is drawn as a new node $a \in A$ in the high-dimensional network, it has a corresponding visible node $a' \in A'$ in the 2-D network. The u and σ are the mean and variance of the weights in the high-dimensional space. The u' and σ' are the mean and variance of the weights in the 2-D space. The weight w_a in the high-dimensional space and the weight w_a' in the 2-D space are normalized.

$$w_a = \frac{w_a - u}{\sigma}, w_a' = \frac{w_a' - u'}{\sigma'}. \tag{17}$$

We introduce S to achieve the goal of preserving the relative distance. S is the standard measurement for measuring the difference between the distance of nodes a, b in the high-dimensional space and the distance of corresponding visible nodes a', b' in the 2-D space.

$$S(a,b) = (d(w_a - w_b) - d(w_a' - w_b'))^2. \tag{18}$$

where d is the Euclidean distance.

To preserve relative distance for all the nodes, we use S_1 to keep the topological network structure of the high-dimensional space to the 2-D space.

$$S_1(a) = \sum_{b\in A} S(a,b). \tag{19}$$

During the process of training, the weights vector of visible nodes is modified to minimize S_1. In each iteration, for each node $a \in A$, the weight vectors of corresponding visible nodes a' in the 2-D network w_a' is updated as follows:

$$w_a' \leftarrow w_a' - \eta_1 \frac{\partial S_1(a)}{\partial w_a'}, \tag{20}$$

where η_1 is positive learning rate which is set as 0.5. We set the number of iterations as 500.

With the analysis, we present how to construct the high-dimensional network and its corresponding 2-D network part of V-SOINN in Algorithm 1.

3.2. Step 2: Classify nodes to different classes

All high-dimensional nodes are initialized as unclassified. For each unclassified node $a \in A$, we label it as a new class L_a, then find all the unclassified nodes b that edges $age_(a,b) \in C$ and label it as

the same class as node a. The visible nodes are labeled as the same class as their corresponding high-dimensional nodes.

We present how to classify the nodes to different classes in V-SOINN in Algorithm 2.

Algorithm 2. Step 2: Classify nodes to different classes.

```
Intialization: set a ∈ A unclassified
repeat
  input:a ∈ A
  if a unclassified then
    a belongs to a new class P
    repeat
      if (a, b) ∈ C then
        set b belongs to class P
      end if
    until All connections in C are searched.
  end if
until All nodes in A are trained.
```

3.3. Step 3: Map the high-dimensional data into the visible network

In step 1, the visible nodes are drawn in the 2-D space. We construct the visible network and map the high-dimensional nodes to it. This step is similar to the network learning process, but largely simplified. The visible network structure remains unchanged during this step. The sizes of the visible nodes in the 2-D space are updated.

For each input data point x, we finds its nearest node s_1 in the high-dimensional network using Eq. 11. Then the weight of s_1 in the high-dimensional space w_{s_1}, is updated by using Eq. 15. By this means, the point x is mapped to the visible node s'_1 in the 2-D space, and the visible node is activated once. The activation count of s'_1 is increased by 1. After all the data points are processed, the size of each visible node is proportional to its activation count in the 2-D space. The distribution of the input data is shown in the low-dimensional space. The purpose of visualizing the high-dimensional data in the low-dimensional space is achieved.

How to map the high-dimensional data into the visible network in V-SOINN is given in Algorithm 3.

Algorithm 3. Step 3: Map the high-dimensional data into the visible network

```
Initialization: set the activation count of all the visible nodes
  in the 2-D space count_{a'} = 0
repeat
  input:x ∈ R^n
  s_1 = arg min_{a∈A} ||x − w_a||
  set count_{s'_1} = count_{s'_1} + 1
until All nodes in A are trained.
Draw visible nodes in the 2-D space. The size of each visible
  node is in direct proportion to its activation count.
```

4. Experiment

In this section, we present the visualization effects of V-SOINN from two aspects. We measure the effectiveness of keeping topological structure by different methods, i.e, PCA, t-SNE, MDS, SOM, NG and V-SOINN. Then we analyze the V-SIONN classification results in the visible network. We train a model completely. The nodes that belong to the same class in the training dataset are drawn in the same color. The original NG algorithm learns the topological structure of dataset in the feature dimension. We combine NG with MDS to visualize the learned the topological structure in the 2-D space. The results are showed through some benchmark datasets, i.e., an artificial dataset with 3 dimensions, the Iris dataset with 4 dimensions, and the MNIST digit dataset with 784 dimensions. For all models, we set output space dimension to 2.

4.1. The effectiveness of keeping topological structure

To compare the quality of different visualization methods, it is necessary to measure the quality. In this paper, we use Relative Standard Deviation (RSD) to measure the effectiveness of maintaining topology. The essence of RSD is judging if the distances between an input data point and its neighboring data points in the high-dimensional space are proportional to those in the low-dimensional output space. Firstly, the distances between any input data point and its k nearest neighbor data points in the input space are calculated. Then we compute the distances between the corresponding data nodes in the 2-D space. After that, the ratios of all distances between data points and their neighboring ones in the input space to the distances in the 2-D space are computed. Then the mean of the ratios μ can be obtained, as well as the standard deviation σ. The RSD is computed as $RSD = \frac{\sigma}{\mu}$. In an ideal model, the ratios of all points are equal, so RSD must be zero. In the real world, the closer the RSD is to zero, the better the performance of the model.

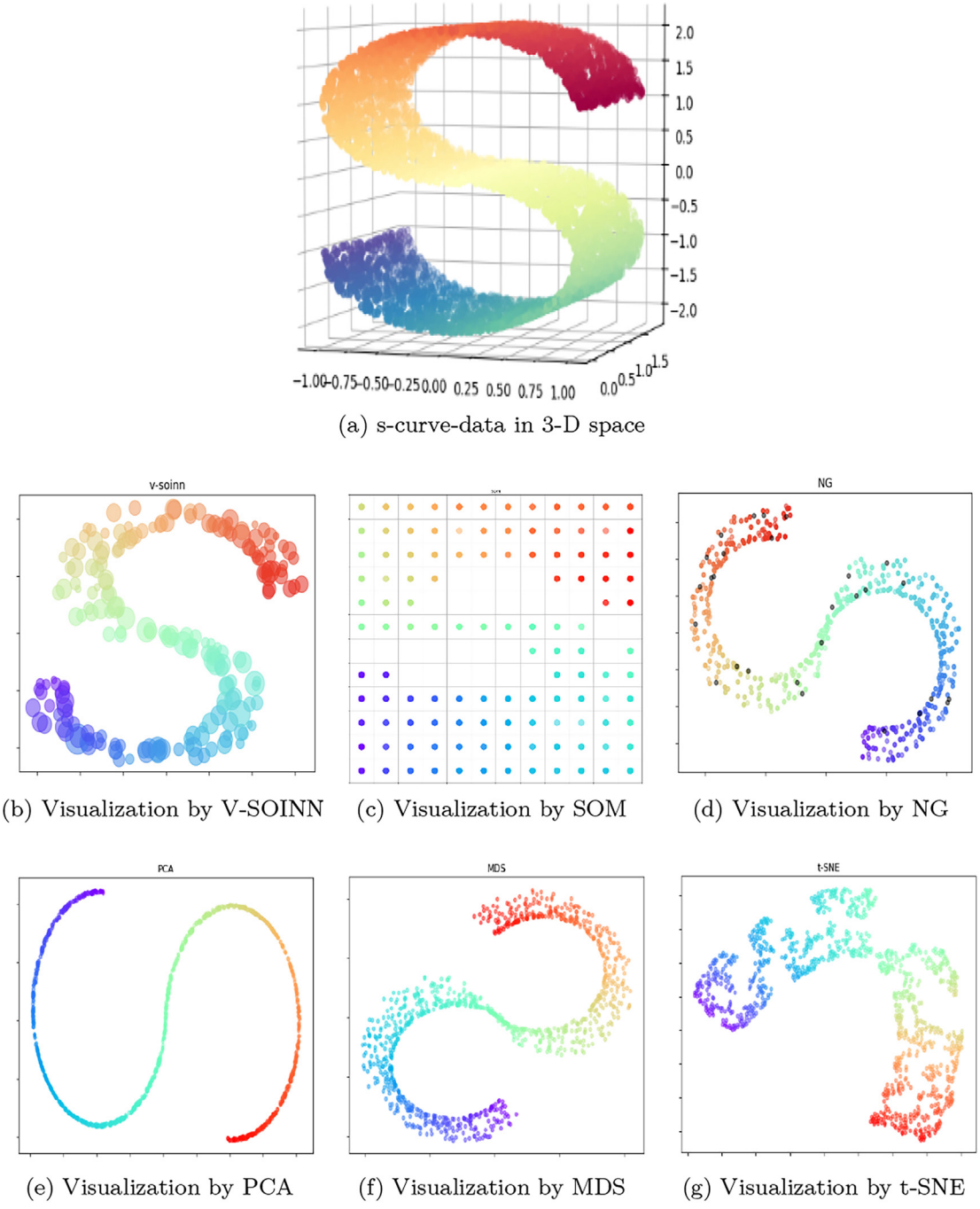
In this paper, the value of k in the RSD is two. For PCA, t-SNE, MDS and V-SOINN, RSD is calculated from the input data and their projection nodes. For SOM, NG, RSD is computed by the training weights of neurons in the input space and the output space. All experiments were implemented in Python 3.7 on a PC.

4.1.1. Artificial dataset

In this experiment, we do visualization on an artificial dataset call "s-curve-data". As we can see from Fig. 1 (a), s-curve-data dataset is made up of 1000 points. These points lie on a three-dimensional space. We express this dataset in the 2-D space with different methods.

For PCA, the number of components to keep is set as 2 which means project the high-dimensional data to the 2-D space. For MDS, the number of neighbors is 20. The algorithm will be started with 4 different initialization. The best result (the one with the lowest RDS) will be selected. For t-SNE, the perplexity is the number of nearest neighbors that is used in the learning algorithms. The perplexity is set as 30. For SOM, the size of map in the 2-D space is set as 12×12. The learning rate is 0.5. The number of iterations is 100. All the 1000 samples in the dataset are used sequentially each iteration. For NG, the number of neural nodes is set as 500. The adaptation step size is 2.5 and the neighborhood range is 2.5. For V-SOINN, the noise elimination period is set as 500 and the map in the 2-D space is generated automatically.

We show the visualization result of the s-curve-data dataset in Fig. 1. The size of the neural nodes in V-SOINN is in direct proportion to the count of being activated. The distribution of input data in the low-dimensional space can be obtained by the size of nodes. As shown in Fig. 1, the shape of dataset can be kept in the low-dimensional space by V-SOINN. There are some inactivated nodes in the figure visualized by NG marked in black, which makes the shape of the dataset not well maintained. The number of inactivated nodes is 30. The visualization result generated by t-SNE is hard for humans to explain its structure in the high-dimensional space. Compared with t-SNE, V-SOINN is easier for people to understand the structure of dataset.



(a) s-curve-data in 3-D space

(b) Visualization by V-SOINN (c) Visualization by SOM (d) Visualization by NG

(e) Visualization by PCA (f) Visualization by MDS (g) Visualization by t-SNE

Fig. 1. Visualization of s-curve-data dataset.

Table 1 shows RSD results based on s-curve-data dataset. As we can see from Table 1, MDS gets the best result. It is because MDS is designed to keep the euclidean distance between pairwise points. It has a good performance for the datasets distributed in the high-dimensional space evenly. The t-SNE, PCA and NG do not perform well in topology preserving on the s-curve-data dataset. The RSD value of SOM is slightly larger than V-SOINN. The RSD value of V-SOINN is smaller than any other methods' RSD values except MDS.

4.1.2. Iris dataset

In this experiment, we do visualization on a real dataset called "Iris". The Iris dataset is a 4-D dataset consisting of three classes. The number of data in each class is 50. The Iris dataset is widely used for data visualization. We express this dataset in the 2-D space with different methods.

For PCA, the number of components to keep is set as 2 which means project high-dimensional data to 2-D space. For MDS, the number of neighbors is 20. The algorithm will be started with 4 different initialization. The best result (the one with the lowest RDS) will be selected. For t-SNE, the perplexity is set as 10. For SOM, the size of map in the 2-D space is set as 7×7. The learning rate is 0.5. The number of iterations is 100. All the 150 samples in the dataset are used sequentially each iteration. For NG, the number of neural nodes is set as 150. The adaptation step size is 2.5 and the neigh-

Table 1
Comparison of methods by using RSD on s-curve-data dataset.

V-SOINN	SOM	NG	PCA	MDS	t-SNE
0.947	1.254	3.307	3.176	0.582	3.547

borhood range is 2.5. For V-SOINN, the noise elimination period is set as 50 and the map in the 2-D space is generated automatically.

We show the visualization result of the Iris in Fig. 2. Nodes in different colors represent different categories in the figures. The size of the node is proportional to its activation count in V-SOINN. We can clearly see the distribution of input data in the 2-D space by V-SOINN. The larger the node, the more input data points are projected on it. The visualization result generated by SOM can not express the distribution of input data. In addition, we can see the classification result of the nodes. As shown in Fig. 2, the different classes can be well classified in the 2-D space by V-SOINN. All three classes can be distinguished clearly. The visualization results generated by NG, PCA, MDS and t-SNE do not well separate all nodes in different classes. There are some inactivated nodes marked in black in the figure visualized by NG. The number of inactivated nodes is 30. These nodes are noisy nodes in the classification result.

Table 2 shows RSD results based on the Iris dataset. As we can see from Table 2, NG gets the worst result. The RSD value of V-SOINN is smaller than any other methods' RSD values.

Combining results of Fig. 2 and Table 2, we can see that V-SOINN performs well in keeping the topological structure on the Iris dataset. The density distribution of the nodes can be shown by using the node size in V-SOINN.

4.1.3. MNIST digit dataset

In these experiments, we obtain the visualization results on the "MNIST" digit dataset.

The MNIST digit dataset is a 784-D dataset consisting of ten classes. The experiments were repeated six times with number of samples 100, 500, 1000, 5000, 10000 and 60000. We express this dataset in the 2-D space with different methods.

For PCA, the number of components to keep is set as 2 which means project high-dimensional data to 2-D space. The number of neighbors is 20 in MDS. The algorithm will be started with 4 different initialization. The best result (the one with the lowest RDS) will be selected. For t-SNE,the perplexity is set as 5, 15, 20, 30, 50 and 50 in order. The size of SOM map is set as 7×7, 10×10, 12×12, 15×15, 20×20, 25×25 in order in these experiments. The learning rate is 0.5. The number of iterations is 100. For NG, the number of neural nodes is set as 50, 100, 500, 500, 1000 and 5000 in order. The adaptation step size is 2.5 and the neighborhood range is 2.5. The noise elimination period of V-SOINN is 50, 250, 500, 500, 500, 500 in order. Here we show the visualization results with 5000 samples generated by each method.

Nodes in different colors represent different categories in the figures. The size of the node is proportional to its activation count in V-SOINN. As shown in Fig. 3, we can see the density distribution of each class by the size of nodes. In Fig. 3, we can see the classifi-

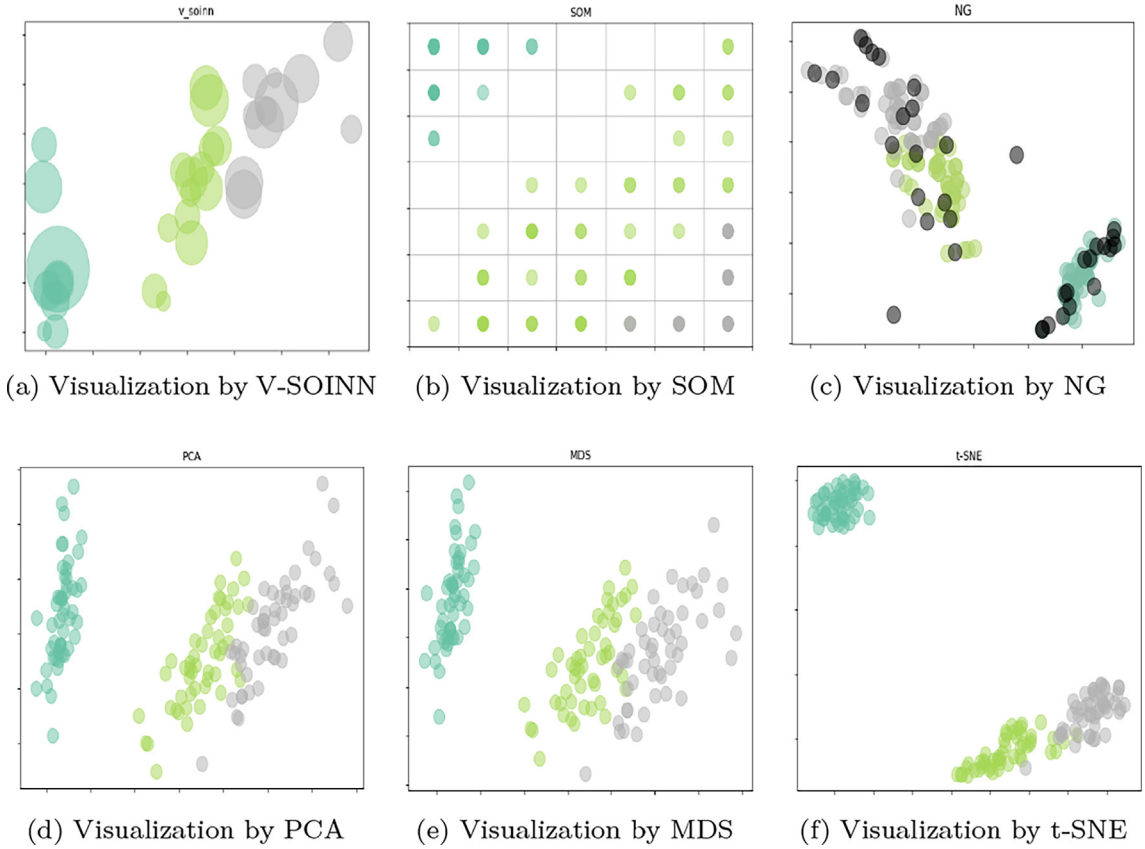


Fig. 2. Visualization of the Iris dataset.

Table 2
Comparison of methods by using RSD on the Iris dataset.

V-SOINN	SOM	NG	PCA	MDS	t-SNE
0.752	1.158	2.701	1.278	1.208	2.115

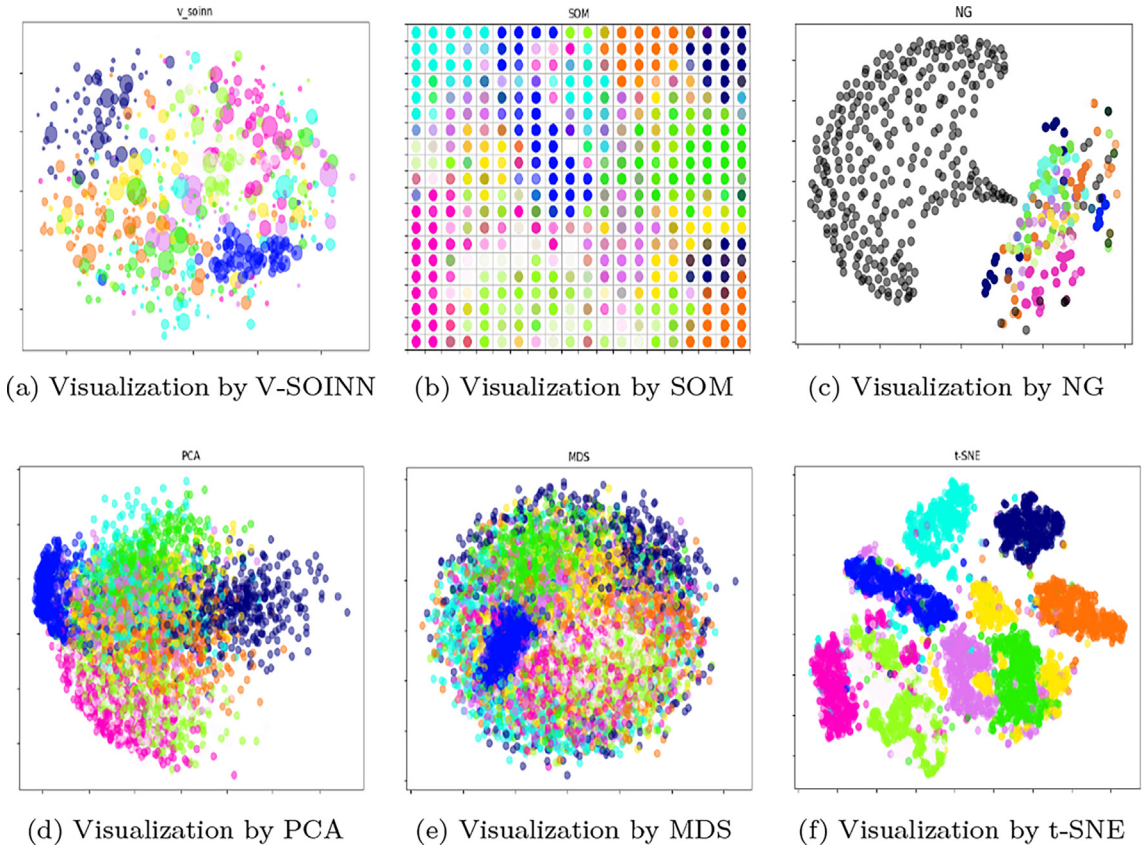


Fig. 3. Visualization of the MNIST digit dataset.

cation result of the nodes. The classes which are easy to classify are distinguished in the 2-D space by V-SOINN. There are many inactivated nodes marked in black in the figure visualized by NG. These nodes are noisy nodes in the classification result. They have a bad influence on the visualization result.

Table 3 shows the RSD results based on the MNIST digit dataset with different sample numbers. In each experiment, the RSD value of V-SOINN is smaller than other methods. The RSD value of t-SNE is too large to keep the topological structure. In different experiments, as the number of samples increases, RSD value of V-SOINN grows at a slow pace.

Combining the results of Fig. 3 and Table 3, we can see that V-SOINN can keep the topological structure stably. V-SOINN can keep topological structure well on the MNIST digit dataset with different number of samples. The density distribution of the nodes can be showed in V-SOINN as well.

4.2. Classification results in the visible network

We have presented the visualization effects of V-SOINN compared with those of PCA, t-SNE, MDS, and SOM. We can see that V-SOINN can preserve the topological structure of the input data. V-SOINN also can show classification results while PCA, t-SNE, MDS, and SOM can not. In this section, we show the number of classes that V-SOINN learned in the 2-D space. Results are shown through the Iris dataset and the MNIST digit dataset. The experimental conditions are the same as the above. We perform 10 times training for V-SOINN and record the average classes numbers.

Table 3
Comparison of methods by using RSD on the MNIST digit dataset.

Sample numbers	V-SOINN	SOM	NG	PCA	MDS	t-SNE
100	0.445	0.925	0.904	0.811	0.712	0.821
500	0.784	1.150	0.871	2.359	1.801	2.482
1000	1.336	1.478	3.699	2.07	1.646	2.394
5000	1.010	1.748	1.124	2.028	2.018	4.006
10000	1.253	1.702	1.666	1.824	4.016	8.719
60000	1.316	1.587	2.375	1.745	3.874	10.479

Table 4
Number of classes for V-SOINN on benchmark datasets.

Dataset	Sample numbers	Number of true classes	Number of classes for V-SOINN
Iris	150	3	3
MNIST	100	10	8.7
MNIST	500	10	8.5
MNIST	1000	10	12.3
MNIST	5000	10	11.2
MNIST	10000	10	10.8
MNIST	60000	10	10.5

Table 4 shows the number of classes for V-SOINN on the benchmark datasets. We use V-SOINN to classify test data into different classes. For the Iris dataset, V-SOINN can always calculate the correct classes numbers. For the MNIST dataset, the number of classes varies from 8.5 to 12.3 for different sample sizes. We can see that V-SOINN can calculate the number of classes in a reasonable error range.

5. Conclusion

In this paper, we propose a new visualization method, V-SOINN. This algorithm constructs a topology preserving visible network automatically and classifies visible nodes to different classes in the low-dimensional space. V-SOINN can learn the low-dimensional visible nodes incrementally while preserving the relative distance between the input data points. V-SOINN can well depict the density distribution of datasets. V-SOINN is able to show the number of classes in the low-dimensional space as well. Compared to SOM, V-SOINN gets neurons automatically and is free of prior condition. The structure of neurons changes according to the input data. Experiments show that the topology preserving ability of V-SOINN is good and V-SOINN has the classification ability to classes which are easy to classify. Both the topology preserving ability and the classification ability are considered, V-SOINN is an effective approach for visualization of multidimensional data.

CRediT authorship contribution statement

Hui Dou: Methodology, Software, Writing - original draft. **Baile Xu:** Validation, Writing - review & editing. **Furao Shen:** Conceptualization, Supervision. **Jian Zhao:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant Nos. (61876076).

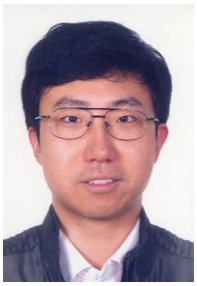
References

[1] S. Liu, D. Maljovec, B. Wang, P.T. Bremer, V. Pascucci, Visualizing high-dimensional data: advances in the past decade, IEEE Trans. Visual Comput. Graphics 23 (3) (2017) 1249.
[2] K. Pearson, Liii. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philos. Mag. J. Sci. 2(11) (1901) 559–572..
[3] W.S. Torgerson, Multidimensional scaling: I. theory and method, Psychometrika 17 (4) (1952) 401–419.
[4] J. Aurisano, J. Hwang, A. Johnson, L. Long, M. Crofoot, T. Berger-Wolf, Bringing the field into the lab: large-scale visualization of animal movement trajectories within a virtual island, in: 2019 IEEE 9th Symposium on Large Data Analysis and Visualization (LDAV), 2019, pp. 83–84.
[5] J.B. Tenenbaum, V. De Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[6] N. P. Belkin M, Laplacian eigenmaps and spectral techniques for embedding and clustering, Adv. Neural Inf. Process. Syst. (2001) 585–591..
[7] L.V. Der Maaten, G.E. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[8] D.A. Keim, M. Sips, M. Ankerst, Visual data-mining techniques, Visualiz. Handb. (2005) 831–843.
[9] H. Chen, S. Zhang, W. Chen, H. Mei, J. Zhang, A. Mercer, R. Liang, H. Qu, Uncertainty-aware multidimensional ensemble data visualization and exploration, IEEE Trans. Visual Comput. Graphics 21 (9) (2015) 1072–1086.
[10] R.A. Pupin de Oliveira, L.F. Silva, D.M. Eler, Hybrid visualization: a new approach to display instances relationship and attributes behaviour in a single view, in: 2015 19th International Conference on Information Visualisation, 2015, pp. 277–282..
[11] T. Yamaguchi, T. Ichimura, Visualization using multi-layered u-matrix in growing tree-structured self-organizing feature map, in: IEEE International Conference on Systems, 2011..
[12] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[13] B. Guo, L. Song, T. Zheng, H. Liang, H. Wang, A comparative evaluation of som-based anomaly detection methods for multivariate data, in: 2019 Prognostics and System Health Management Conference (PHM-Qingdao), 2019, pp. 1–6.
[14] T. Martinetz, K. Schulten, et al., A neural-gas network learns topologies (1991)..
[15] S. Furao, T. Ogura, O. Hasegawa, An enhanced self-organizing incremental neural network for online unsupervised learning, Neural Netw. Off. J. Int. Neural Network Soc. 20 (8) (2007) 893–903.
[16] S. Furao, O. Hasegawa, An incremental network for on-line unsupervised classification and topology learning, Neural Networks 19 (1) (2006) 90–106.
[17] P. Hartono, Y. Take, Pairwise elastic self-organizing maps, in: International Workshop on Self-organizing Maps and Learning Vector Quantization, 2017..

Hui Dou received the bachelar's degree from Beijing Jiaotong University. She is currently a Ph.D. student at Robotic Intelligence & Neural Computing Group, Nanjing University. Her research interest is artificial neural networks.

Baile Xu received the bachelar's degree from Department of Software Engineering, Shandong University. Currently he is a Ph.D. student at Nanjing University. His research interests include artificial neural networks and machine learning.

Furao Shen received the Engineering degree from Tokyo Institute of Technology, Tokyo, Japan, in 2006. Currently he is a professor at Nanjing University, China. His research interests include neural computing and robotic intelligence.



Jian Zhao received the B.S. degree from Nanjing University, Nanjing, China, the M.Sc. degree from Hamburg University of Technology, Hamburg, Germany, and the Dr. Sc. degree in electrical engineering from Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. From 2010 to 2015, he was a Research Scientist with the Institute for Infocomm Research, A*STAR, Singapore. Currently, he is an Associate Professor with the School of Electronic Science and Engineering in Nanjing University. His research interests include deep neural networks, mathematical optimization, and wireless communication networks.